0003 – Aprendiendo de QA

Autor:
Mariela Vaniglia – QA Lead

Disertante:
Yesica S Sosa – Analista QA Sr

Lagash

/ TABLA DE CONTENIDOS

1 / RESUMEN

En este documento se van a poder encontrar los contenidos correspondientes a la tercer clase de Lagash University (0003).

Como herramienta de comunicación, en este curso vamos a utilizar Slack. La url del mismo es: https://lagash-university.slack.com

2 / QA

2.1 Definición

La calidad de los productos y servicios de una organización está determinada por la capacidad para satisfacer a los clientes, y por el impacto previsto y el no previsto sobre las partes interesadas pertinentes. [ISO 9000:2015]

Calidad es un concepto subjetivo. Está relacionado con las percepciones y expectativas de cada individuo para comparar un objeto con cualquier otro de su misma especie. Diversos factores influyen directamente en esta definición. En particular, la calidad del servicio prestado por una determinada empresa es asociado a su cualidad en relación a la percepción de satisfacción. La calidad de un producto en general se refiere a la cualidad y durabilidad del objeto en cuestión.

El Aseguramiento de la Calidad, más conocido como Quality Assurance (QA), es un conjunto de actividades aplicadas bajo un sistema de Gestión de la Calidad que persigue como objetivo que los requisitos de calidad de un producto o servicio sean satisfechos.

2.2 Proceso de QA

QA implica control a nivel metodologías, procesos y código. Es análisis, diseño, seguimiento, control, planificación y otras tareas que resultan esenciales en cualquier proyecto. El enfoque a aplicar en cada caso varía dependiendo de la estructura de la organización, la importancia del software, los riesgos, los costos y tecnologías disponibles.

El objetivo principal de un proceso de QA es asegurar que un sistema funcionará correctamente. Por eso es importante que esta actividad se realice en todas las etapas, no al final del proyecto.

El aseguramiento de calidad aplica en todos los niveles:

	QA Quality Assurance	QC Quality Control	TESTING
OBJETIVO	Desarrollar procesos que eviten errores en el producto basados en estándares de Calidad.	Asegurar que el producto cumpla con los requerimientos y especificaciones antes de su implementación a Producción.	Identificar y solucionar errores de software
APLICA A NIVEL	Procesos	Producto	Diseño de la solución y código Fuente
ORIENTADO A	Prevención	Verificación	Detección

RESPONSABLES DE APLICARLO	Organización, todas las áreas intervinientes	Equipo de Desarrollo	Desarrolladores y testers
CUANDO SE IMPLEMENTA	A lo largo del proceso completo	Previo a la implementación en Producción	A lo largo del proceso de desarrollo o en la etapa de Testing

2.3 Equipo de QA

En Lagash contamos con un equipo de QA, son los encargados de diseñar un set de operaciones (casos de prueba) que validen que el producto cumple con los criterios de aceptación de un requerimiento y que las funcionalidades pre-existentes se mantienen sin alteraciones.

Para ello, llevan a cabo las siguientes tareas principales:

- Análisis de requerimientos
  - Lectura
  - Entendimiento detallado del requerimiento
  - Impacto en funcionalidades pre-existentes
  - Evaluación de los criterios de aceptación
  - Identificación de flujos funcionales no considerados
  - Identificación de conceptos ambiguos

- Diseño de Casos de Prueba - *test cases*
  - Escritura de los casos de prueba en la herramienta de Test Management
  - El set de Casos de Prueba conforman el Plan de Pruebas – *Test Plan*

- Identificación de Precondiciones - *preconditions*
  - Generación de datos previos a la ejecución
  - Configuraciones de ambiente

- Ejecución de Casos de Prueba
  - Ejecución individual de cada caso
  - Tracking de resultado obtenido
  - Registración de defectos - *issues*
  - Seguimiento de defectos

- Liderazgo de instancias de pruebas funcionales con el cliente
  - Dependiendo las características del proyecto, liderar las actividades de Pruebas de Aceptación con el cliente.

- Reportes y métricas
  - Obtención de métricas y feedback

- Soporte al Negocio
  - Asistencia al equipo de Producto respecto a las funcionalidades implementadas

3 / TESTING

3.1 Definición

Testing es una actividad que se realiza durante todas las fases del ciclo de vida de desarrollo. Está relacionada con la planificación, preparación y evaluación del producto de software a fin de determinar el cumplimiento de requerimientos específicos y de identificar defectos (ISQTB)

3.2 Tipos de Testing

Según el criterio, existen distintas clasificaciones. Las más importantes son:

- Por ejecución:
  - Manual
  - Automatizado

- Por enfoque
  - Pruebas de Caja Blanca
  - Pruebas de Caja Negra

- Por objetivo
  - Funcionales: *lo que el sistema hace*
    - Pruebas Unitarias
    - Pruebas de Integración
    - Pruebas de Humo
    - Pruebas de Aceptación (UAT)
    - Pruebas de Regresión

  - No funcionales: *cómo trabaja el sistema*
    - Pruebas de seguridad
    - Pruebas de Stress
    - Pruebas de Performance
    - Pruebas de Carga
    - Pruebas de Compatibilidad

Pruebas Manuales – *Manual Testing*

Es el proceso de ejecución de casos de prueba en forma manual. Requiere un analista de QA que actue de usuario final para asegurar que el producto tiene el comportamiento deseado.

Pruebas automatizadas – *Automated testing*

Es un método basado en herramientas de software que simulan la ejecución manual y comparan resultados en pos de la predicción.

Pruebas de Caja Blanca – *Whitebox testing*

Son pruebas realizadas sobre el código fuente de la aplicación.

Básicamente, el analista-tester selecciona distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y asegurarse que se devuelven los valores de salida adecuados. Al estar basadas en una implementación concreta, si ésta se modifica, las pruebas también deberán rediseñarse.

Pruebas de Caja Negra – *Blackbox testing*

Es una técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software.

Se enfoca solamente en las entradas y salidas del sistema, sin tener conocimiento de la estructura interna del programa de software. Para obtener el detalle de cuáles deben ser esas entradas y salidas, nos basamos en los requerimientos de software y especificaciones funcionales.

Pruebas Unitarias – *Unit testing*

Es una forma de comprobar el correcto funcionamiento de una unidad de código. Por ejemplo en diseño estructurado o en diseño funcional una función o un procedimiento, en diseño orientado a objetos una clase.

Sirve para asegurar que cada unidad funcione correcta y eficientemente por separado. Además de verificar que el código hace lo que tiene que hacer, verificamos que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve. Si el estado inicial es válido entonces el estado final es válido.

Pruebas de Integración – *Integration testing*

Estas pruebas se realizan una vez que todos los elementos unitarios que componen el software, funcionan juntos correctamente probándolos en grupo. Se centra principalmente en probar la comunicación entre los componentes y sus interacciones.

Pruebas de humo – *Smoke testing*

Estas pruebas se realizan sin entrar en detalle sobre las funcionalidades. Brindan una cobertura amplia pero poco profunda.

Pruebas de Regressión – *Regresion testing*

Apuntan a asegurar que el producto continúa respondiendo correctamente más allá del código nuevo que se ha introducido. Se refiere a la revisión de funcionalidades preexistentes.

Pruebas de Aceptación – *User Acceptance testing*

Son pruebas que se realizan con el usuario para que éste determine si un sistema satisface los criterios de aceptación y en función de los resultados, si acepta o no el producto.

Es una instancia fundamental para asegurar el éxito de la implementación final de un proyecto.

Pruebas de Seguridad – *Security testing*

Buscan medir la confidencialidad, integridad y disponibilidad de los datos obtenidos a partir del uso de la aplicación. Permiten identificar amenazas y riesgos relacionados a la seguridad de la información.

Pruebas de Stress – *Stress testing*

Estudia la robustez y confiabilidad del software sometiéndolo a condiciones de uso extremas.

Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

Pruebas de Rendimiento – *Performance testing*

Buscan determinar cuan rápido un sistema realiza una tarea en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad tales como la escalabilidad, fiabilidad y uso de los recursos.

Pruebas de Carga – *Load testing*

Es el tipo más sencillo de pruebas de rendimiento. Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación. Si la base de datos,

el servidor de aplicaciones, etc.. también se monitorizan, entonces esta prueba puede mostrar el cuello de botella en la aplicación.

Pruebas de Compatibilidad – *Load testing*

Las pruebas de compatibilidad son pruebas que verifican el funcionamiento del sistema en diferentes entornos. Por ejemplo se puede probar un sistema en los distintos navegadores o en distintas versiones de máquina virtual o en distintos sistemas operativos o en distintos dispositivos. El objetivo es garantizar el correcto funcionamiento de la aplicación en todos aquellos entornos en los que tiene que funcionar de forma correcta.

4 / ISSUES

4.1 Definiciones

Error: una equivocación de una persona al desarrollar alguna actividad de desarrollo de software.

Defecto: se produce cuando una persona comete un error. (por ej: cuando un diseñador comprende mal un requerimiento y crea un diseño inadecuado).

Falla: es un desvío respecto del comportamiento esperado del sistema. Puede producirse en cualquier etapa.

Defecto es una vista interna, lo ven los desarrolladores. Falla es una vista externa, la ven los usuarios. No todas las fallas provienen de defectos ni todos los defectos terminan en una falla. Los defectos del software son esencialmente diferentes de los de hardware: los materiales se fatigan, se desgastan.

Mejoras: son necesidades no identificadas por el usuario durante el proceso de relevamiento.

4.2 Registración de bugs

Los bugs se registran en la herramienta para Bug Tracking definida en el proyecto y deben proveer la siguiente información:

- ID o Título
- Severidad (critica, alta, media, baja, mejora)
- Breve descripción
- Pasos para reproducir el error
- Precondiciones a tener en cuenta
- Resultado esperado

- Resultado obtenido con screenshots
- Datos exactos (si fuera el caso)
- Responsable de solucionarlo (owner)
- Ambiente en el cual se detectó el bug (QA, UAT, Producción)
- Versión

5 / HISTORIAL DE REVISION

Fecha	Versión	Descripción	Autor
24/04/2018	1.0	Volcado del contenido de la clase al documento.	Mariela Vaniglia
17/05/2019	1.1	Cambio de día en el titúlo de la portada y en algunas páginas (ejemplo, clase 5 por 3)	Yesica Sosa